

- JP 11316675/pn

L1 ANSWER 1 OF 1 JAPIO COPYRIGHT 2001 JPO  
ACCESSION NUMBER: 1999-316675 JAPIO  
TITLE: COMPUTER SYSTEM  
INVENTOR: YOSHIDA KAZUKI; RALPH E JOHNSON  
PATENT ASSIGNEE(S): TOSHIBA CORP)  
PATENT INFORMATION:

PATENT NO	KIND	DATE	ERA	MAIN IPC
-----				
***JP 11316675***		A19991116	Heisei	G06F009-06

JP

APPLICATION INFORMATION

ST19N FORMAT:	JP1999-009330	19990118
ORIGINAL:	JP11009330	Heisei
PRIORITY APPLN. INFO.:	US1998 8218	19980116
SOURCE:	PATENT ABSTRACTS OF JAPAN (CD-ROM), Unexamined Applications, Vol. 99	

INT. PATENT CLASSIF.:

MAIN: G06F009-06

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a computer system for assembling a new or corrected program without inputting, correcting or recompiling a program code.

SOLUTION: This system is provided with a compound display part 108 for displaying, in response to a first user input, at least two first graphics expressing a recycle program component as the unit of data processing stored in a memory on a display unit 102 and displaying, in response to a second user input, at least one second graphic showing the connection relation of at least two recycle program components expressed with at least two first graphics on the display unit 102. This system is also provided with a recycle component selecting part 107 for selecting any recycle program component in response to the first input and generating a first request, and a recycle component connecting part 110 for defining the connecting relation of the selected recycle program component and generating a second request.

COPYRIGHT: (C)1999,JPO

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-316675

(43) 公開日 平成11年(1999)11月16日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/06

識別記号

5 3 0

F I

G 0 6 F 9/06

5 3 0 W

審査請求 未請求 請求項の数39 O L (全 23 頁)

(21) 出願番号 特願平11-9330

(22) 出願日 平成11年(1999) 1月18日

(31) 優先権主張番号 09/008218

(32) 優先日 1998年 1月16日

(33) 優先権主張国 米国 (US)

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 吉田 和樹

東京都府中市東芝町1番地 株式会社東芝  
府中工場内

(72) 発明者 ラルフ・イー・ジョンソン

アメリカ合衆国、イリノイ州、シャン  
ペイン、ダブリュ・グリーン・ストリー  
ト 708

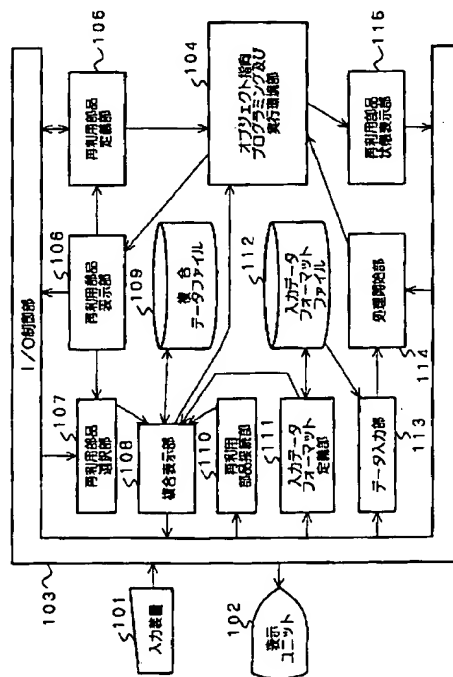
(74) 代理人 弁理士 鈴江 武彦 (外6名)

(54) 【発明の名称】 コンピュータシステム

(57) 【要約】

【課題】 プログラムコードを入力、修正、再コンパイルせずに、新しい、または修正されたプログラムを組み立てるコンピュータシステムを提供する。

【解決手段】 メモリに格納されているデータ処理の単位である再利用プログラム部品を表す少なくとも二つの第1図形を第1のユーザ入力に応答して表示ユニット(102)に表示し、少なくとも二つの第1図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第2図形を第2のユーザ入力に応答して表示ユニット(102)に表示する複合表示部(108)と、第1の入力に応答して再利用プログラム部品を選択し、第1の要求を発生する再利用部品選択部(107)と、選択された再利用プログラム部品の接続関係を定義し、第2の要求を発生する再利用部品接続部(110)とを具備する。



## 【特許請求の範囲】

【請求項 1】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリとを有するコンピュータとユーザとの間のインターフェイスを取る方法において、

前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形を第 1 のユーザ入力に応答して表示するステップと、

前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を第 2 のユーザ入力に応答して表示するステップと、

前記第 1 図形で表わされる少なくとも一つの再利用プログラムに入力データを供給するステップと、

前記第 2 図形で表わされる接続関係にある前記第 1 図形で表わされる少なくとも二つの再利用プログラムに応じて入力データを処理するステップと、

を具備するインターフェイス方法。

【請求項 2】 前記メモリに格納されている再利用プログラム部品のいくつかを示す再利用プログラム部品メニューを第 3 のユーザ入力に応答して表示するステップをさらに具備することを特徴とする請求項 1 記載のインターフェイス方法。

【請求項 3】 前記第 1 のユーザ入力は表示メニューに基づく再利用プログラム部品の選択であることを特徴とする請求項 2 記載のインターフェイス方法。

【請求項 4】 前記第 1 図形の一つはユーザによるデータ入力のためのプログラム部品を示し、前記入力装置によりユーザがデータを入力できるようにウィンドウを表示することを特徴とする請求項 1 記載のインターフェイス方法。

【請求項 5】 前記第 2 図形は表示されている他の再利用プログラム部品によるある再利用プログラム部品の起動を示す第 1 のタイプの第 2 図形を含むことを特徴とする請求項 1 記載のインターフェイス方法。

【請求項 6】 前記第 1 のタイプの第 2 図形は第 1 のタイプの第 2 のユーザ入力により表示されることを特徴とする請求項 5 記載のインターフェイス方法。

【請求項 7】 少なくとも二つの前記第 2 図形が表示され、前記第 2 図形は表示されている他の再利用プログラム部品によるある再利用プログラム部品の属性の参照を示す第 2 のタイプの第 2 図形を含むことを特徴とする請求項 6 記載のインターフェイス方法。

【請求項 8】 前記第 2 のタイプの第 2 図形は第 2 のタイプの第 2 のユーザ入力により表示されることを特徴とする請求項 7 記載のインターフェイス方法。

【請求項 9】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリとを有するコンピュータとユーザとの間のインターフェイスを取る方法において、

前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形を第 1 のユーザ入力に応答して表示するステップと、

前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を第 2 のユーザ入力に応答して表示するステップと、

前記表示されている第 1、第 2 図形に関し、第 1 図形により表わされている再利用プログラム部品と、第 2 図形により表わされている接続関係に応じてデータを処理する処理を定義する処理情報を前記メモリに格納するステップと、

を具備するインターフェイス方法。

【請求項 10】 前記メモリに格納されている再利用プログラム部品のいくつかを示す再利用プログラム部品メニューを第 3 のユーザ入力に応答して表示するステップをさらに具備することを特徴とする請求項 9 記載のインターフェイス方法。

【請求項 11】 前記第 1 のユーザ入力は表示メニューに基づく再利用プログラム部品の選択であることを特徴とする請求項 10 記載のインターフェイス方法。

【請求項 12】 前記第 2 図形は表示されている他の再利用プログラム部品によるある再利用プログラム部品の起動を示す第 1 のタイプの第 2 図形を含むことを特徴とする請求項 9 記載のインターフェイス方法。

【請求項 13】 前記第 1 のタイプの第 2 図形は第 1 のタイプの第 2 のユーザ入力により表示されることを特徴とする請求項 12 記載のインターフェイス方法。

【請求項 14】 少なくとも二つの前記第 2 図形が表示され、前記第 2 図形は表示されている他の再利用プログラム部品によるある再利用プログラム部品の属性の参照を示す第 2 のタイプの第 2 図形を含むことを特徴とする請求項 13 記載のインターフェイス方法。

【請求項 15】 前記第 2 のタイプの第 2 図形は第 2 のタイプの第 2 のユーザ入力により表示されることを特徴とする請求項 14 記載のインターフェイス方法。

【請求項 16】 前記処理情報は再利用プログラム部品として前記メモリに格納されることを特徴とする請求項 9 記載のインターフェイス方法。

【請求項 17】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品と請求項 9 に記載の処理情報とを格納するメモリとを有するコンピュータを用いてデータを処理する方法において、前記処理情報に基づいて、前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形と、前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を表示するステップと、前記第 1 図形で表わされる少なくとも一つの再利用プログラムに入力データを供給するステップと、

前記第2図形で表わされる接続関係にある前記第1図形で表わされる少なくとも二つの再利用プログラムに応じて入力データを処理するステップと、

を具備するデータ処理方法。

【請求項18】 前記第1図形の一つはユーザによるデータ入力のためのプログラム部品を示し、前記入力装置によりユーザがデータを入力できるようにウィンドウを表示することを特徴とする請求項17記載のデータ処理方法。

【請求項19】 データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリを有するコンピュータに実装され、再利用プログラム部品からプログラムを作成する方法において、

第1の入力情報に応じて少なくとも二つの再利用プログラム部品を選択するステップと、

選択された再利用プログラム部品の少なくとも一つの起動を第2の入力情報に応じて定義するステップと、

を具備し、前記選択ステップ、定義ステップはプログラムコードの生成、修正、コンパイルを必要とせずに実行されるプログラム作成方法。

【請求項20】 他の再利用プログラム部品による前記少なくとも一つの再利用プログラム部品の属性の参照を定義するステップをさらに具備し、前記定義ステップはプログラムコードの生成、修正、コンパイルを必要とせずに実行されることを特徴とする請求項19記載のプログラム作成方法。

【請求項21】 定義した起動を表わす情報を前記メモリに格納するステップをさらに具備することを特徴とする請求項19記載のプログラム作成方法。

【請求項22】 選択された再利用プログラム部品を定義した起動に応じて実行するステップをさらに具備することを特徴とする請求項19記載のプログラム作成方法。

【請求項23】 特定の条件が満たされた場合のみ前記第1再利用プログラム部品により第2再利用プログラム部品が起動されるように、起動条件を定義するステップをさらに具備することを特徴とする請求項19記載のプログラム作成方法。

【請求項24】 前記起動条件は前記第1再利用プログラム部品と第2再利用プログラム部品の間に起動条件を表わす条件部品を挿入することにより定義され、第1再利用プログラム部品は条件部品を起動し、条件部品は起動条件が満たされた場合のみ第2再利用プログラム部品を起動することを特徴とする請求項23記載のプログラム作成方法。

【請求項25】 前記第1再利用プログラム部品による複数の第2再利用プログラム部品の起動するための複数の条件を定義するステップをさらに具備し、前記第1再利用プログラム部品はどの条件が満たされているかに応じて複数の第2再利用プログラム部品のいずれかを起動

することを特徴とする請求項19記載のプログラム作成方法。

【請求項26】 前記起動条件は前記第1再利用プログラム部品と第2再利用プログラム部品の間に複数の第2再利用プログラム部品の起動条件をそれぞれ格納する複数の条件部品からなる条件部品セットを挿入することにより定義され、第1再利用プログラム部品は条件部品セットを起動し、条件部品セットはどの条件が満たされているかに応じて複数の第2再利用プログラム部品のいずれかを起動することを特徴とする請求項25記載のプログラム作成方法。

【請求項27】 前記条件は時間に関する条件であることを特徴とする請求項23、または請求項25記載のプログラム作成方法。

【請求項28】 データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリとユーザインターフェースを有し、メモリに格納されている再利用プログラム部品からプログラムを作成するコンピュータとユーザとの間のインターフェイスを取る装置において、

オブジェクト指向プログラミング及び環境実行部と、前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第1図形を第1のユーザ入力にตอบสนองしてユーザインターフェースに表示し、前記少なくとも二つの第1図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第2図形を第2のユーザ入力にตอบสนองしてユーザインターフェースに表示する複合表示部と、

第1の入力にตอบสนองして再利用プログラム部品を選択し、第1の要求を発生する再利用部品選択部と、選択された再利用プログラム部品の接続関係を定義し、第2の要求を発生する再利用部品接続部と、を具備するインターフェース装置。

【請求項29】 前記複合表示部は、オブジェクトを生成し、前記オブジェクト指向プログラミング及び環境実行部内で前記第1、第2の要求に応じて、生成されたオブジェクト間の起動を定義することを特徴とする請求項28記載のインターフェース装置。

【請求項30】 前記メモリに格納されている再利用プログラム部品のいくつかを特定するメニューを表示する再利用プログラム表示部をさらに具備することを特徴とする請求項28記載のインターフェース装置。

【請求項31】 処理すべきデータを入力する入力部をさらに具備することを特徴とする請求項28記載のインターフェース装置。

【請求項32】 再利用プログラム部品の状態を示す再利用部品状態表示部をさらに具備することを特徴とする請求項28記載のインターフェース装置。

【請求項33】 選択した再利用プログラム部品と、該部品間の関係を示す情報を格納する複合データファイルをさらに具備することを特徴とする請求項28記載のイ

ンターフェース装置。

【請求項 34】 前記メモリに格納されている再利用プログラム部品から新たな再利用プログラム部品を作成する再利用プログラム部品定義部をさらに具備することを特徴とする請求項 28 記載のインターフェース装置。

【請求項 35】 選択した再利用プログラム部品間の条件を定義する接続管理部をさらに具備することを特徴とする請求項 28 記載のインターフェース装置。

【請求項 36】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリとを有するコンピュータとユーザとの間のインターフェイスを取るためのコンピュータプログラムを記録しコンピュータ読取り可能な記憶媒体において、前記コンピュータプログラムは前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形を第 1 のユーザ入力に応答して表示するプログラムコードと、

前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を第 2 のユーザ入力に応答して表示するプログラムコードと、

前記第 1 図形で表わされる少なくとも一つの再利用プログラムに入力データを供給するプログラムコードと、  
前記第 2 図形で表わされる接続関係にある前記第 1 図形で表わされる少なくとも二つの再利用プログラムに応じて入力データを処理するプログラムコードと、  
を具備することを特徴とする記憶媒体。

【請求項 37】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリとを有するコンピュータとユーザとの間のインターフェイスを取るためのコンピュータプログラムを記録しコンピュータ読取り可能な記憶媒体において、前記コンピュータプログラムは前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形を第 1 のユーザ入力に応答して表示するプログラムコードと、

前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を第 2 のユーザ入力に応答して表示するプログラムコードと、

前記表示されている第 1、第 2 図形に関し、第 1 図形により表わされている再利用プログラム部品と、第 2 図形により表わされている接続関係に応じてデータを処理する処理を定義する処理情報を前記メモリに格納するプログラムコードと、  
を具備することを特徴とする記憶媒体

【請求項 38】 表示装置と、入力装置と、データ処理の処理の単位を表わす再利用プログラム部品と請求項 9 に記載の処理情報とを格納するメモリとを有するコンピュータを用いてデータを処理するためのコンピュータプ

ログラムを記録しコンピュータ読取り可能な記憶媒体において、前記コンピュータプログラムは前記処理情報に基づいて、前記メモリに格納されている再利用プログラム部品を表す少なくとも二つの第 1 図形と、前記少なくとも二つの第 1 図形で表わされる少なくとも二つの再利用プログラム部品の接続関係を示す少なくとも一つの第 2 図形を表示するプログラムコードと、

前記第 1 図形で表わされる少なくとも一つの再利用プログラムに入力データを供給するプログラムコードと、  
前記第 2 図形で表わされる接続関係にある前記第 1 図形で表わされる少なくとも二つの再利用プログラムに応じて入力データを処理するプログラムコードと、  
を具備することを特徴とする記憶媒体。

【請求項 39】 データ処理の処理の単位を表わす再利用プログラム部品を格納するメモリを有するコンピュータに実装され、再利用プログラム部品からプログラムを作成するためのコンピュータプログラムを記録しコンピュータ読取り可能な記憶媒体において、前記コンピュータプログラムは第 1 の入力情報に応じて少なくとも二つの再利用プログラム部品を選択するプログラムコードと、

選択された再利用プログラム部品の少なくとも一つの起動を第 2 の入力情報に応じて定義するプログラムコードと、

を具備し、前記選択ステップ、定義ステップはプログラムコードの生成、修正、コンパイルを必要とせずに実行されることを特徴とする記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、コンピュータシステムに関し、特に、プログラムコードを修正、または再コンパイルせずに、システムの変更を可能にする改良されたコンピュータシステムに関するものである。

【0002】

【従来の技術】ソフトウェア開発ツールとして、Smalltalk、C++、および Java 等のオブジェクト指向のプログラミング・システム(OOPS)が普及している。オブジェクト指向技術の一つの利点はプログラムコードの再利用を向上させる能力にある。オブジェクト指向プログラムでは、現実世界のオブジェクトはオブジェクトと呼ばれる(「インスタンス」とも呼ばれる)ソフトウェア・エンティティにより表され、現実世界のオブジェクトの状態、および挙動はそれぞれのオブジェクトの属性、および演算として定義される。

【0003】オブジェクト指向のプログラミングはオブジェクトの設計と実装、およびそれらの相互作用の具体的記述から構成される。オブジェクト指向プログラミングシステムはオブジェクトやクラス、および継承や多相性等の概念を使用することが特徴である。オブジェクトはデータとそのデータの演算(メソッドと呼ばれること

が多い)との離散的で、独立した(self-contained)組み合わせである。クラスとはオブジェクトの集合に対してその特徴を抽出したものである。この集合内の全てのオブジェクトは特別なクラスに属する、あるいはそのクラスにより生成されることが出来る。用語「継承」は、既存のクラスから新たなクラスを派生させることによりその新クラスを定義するオブジェクト指向プログラミング言語の機構を言う。この派生したクラスは元のクラスのサブクラスと呼ばれ、この元のクラスは派生したクラスのスーパークラスと呼ばれる。その派生したクラスは既存のクラスから全てのデータとメソッドを継承しているが、派生したクラスにはデータまたはメソッドを新たに付加しても良いし、元のクラスの挙動とは異なる挙動をするように、継承したメソッドを再定義しても良い。オブジェクト指向プログラミングでの「多相性」という用語は、異なる挙動を有する同一名のメソッドを指し、この挙動はこれらのメソッドが参照するオブジェクトの種類により異なるものである。メソッドが起動されると、多相性は問題のオブジェクトをラン・タイムで判断することにより適切な演算が実施されることを確実にする。継承はコード共有を促進し、多相性は演算とオブジェクトのラン・タイムの結合を可能にする。

【0004】オブジェクト指向技術はビジネス・ソフトウェアのフレームワークを開発するために使用される。フレームワークは会計システム、データベース、グラフィック・ユーザ・インターフェース、オペレーティングシステム、在庫管理、物流、人的資源管理等の特定の分野でのオブジェクト間の相互作用を記述する抽象的な設計である。例えば、会計フレームワークは会計分野のアプリケーションに共通の知識を含んでいる。そのフレームワークのユーザはそれを使って、特定の会計システムの下位レベルの細部の設計に関わる特定業務用の会計システムを設計しても良い。会計システム用のその様なフレームワークの一例としては、ACCOUNTSソフトウェアがあり、その初期版はポール・D・キーファー(Paul D. Keefer)の論文「会計システム用のオブジェクト指向フレームワーク」に記載されている。これはイリノイ大学アーバナ/シャンペン(Urbana-Champaign)校の大学院へ1994年に提出された修士論文である。

【0005】ACCOUNTSは、コンピュータ化された業務取引処理システムを構築するためにオブジェクト指向プログラミング・システムに実装されたフレームワークである。このフレームワークの下では、ユーザは勘定と取引を定義するためにそのACCOUNTSシステムを使用する。取引は勘定に転記されたタイム・スタンプ済みレコードと考えられる。各勘定はそれに影響を与える取引の履歴を保管する。勘定は、例えば、在庫仕訳帳や、銀行口座や、雇用者レコード等であっても良く、一方、取引は、例えば、仕入と販売取引や、預金と引出や、タイム・カードと給料等であっても良い。勘定は、

そこに記憶された取引から現行総計(running total)を計算するような属性の集合を有していても良い。例えば、在庫勘定は、売残り、当月売上、および当月売上げ商品の原価等の属性を有していても良く、雇用者レコードはこの支払期間にした残業時間、到来給与額、時間給、および有給休暇等の属性を持っていても良い。特定のACCOUNTSシステムでは、勘定、取引と勘定属性、およびそれらの関係は「業務規則」の集合として定義されている。例えば、ある業務規則では、売上げ商品量を適切な在庫勘定に登録し、受け取った代金を現金勘定に登録することにより処理されるものとして販売取引を定義しているかもしれない。

【0006】特定業務用の新しいAccountsシステムは、そのシステムの勘定と取引、および各勘定の属性、更にそれらの勘定により取引を処理する方法を定義することによって設計される(「Accounts」と言う用語は、本明細書ではAccountsフレームワークを実装するソフトウェアと、特定業務用にユーザが設計した特定のAccountsシステムの両方を指す)。Accountsシステムがセットアップされれば、ユーザはそれを使用して、例えば取引を生成したり、勘定属性に問い合わせを行うこともできる。取引は、買掛金システムまたは給与システム用のチェック・ライター等の他のコンピュータシステムから来ても良いし、ユーザが入力画面上で記入することにより作成しても良い。問合せは、簡単なグラフィック・ユーザ・インターフェイス(GUI)ツールや、銀行の報告書や毎月の売上報告等や、問合せを使用して作成する小切手の金額を決定するチェック・ライター等の取引を生成するコンピュータプログラムから行っても良い。

【0007】

【発明が解決しようとする課題】このように、通常、エンドユーザは取引を作成し、属性を問い合わせることによりACCOUNTSシステムと対話する。しかし、業務用のACCOUNTSシステムをセットアップする場合、または業務規則の変化に応じてACCOUNTSシステムを変更する場合は、ユーザが、例えば取引、勘定または属性の定義を追加または変更することが必要になる場合もある。この種の対話は、一般的に、プログラミングにより実現されるが、その際にユーザがプログラムコードを入力または修正したり、そのコードを再コンパイルして実行可能プログラムを生成する必要がある。

【0008】本発明の目的は、プログラムコードを入力、修正、または再コンパイルせずに、新しい、または修正されたプログラムを組み立てるコンピュータシステムに関するものである。さらに、好適にはグラフィック・ユーザ・インターフェースを利用して新しい、または修正されたプログラムを組み立てるコンピュータシステムを提供することが望ましい。また、業務規則の変化を時間とともに追跡可能で、所定の時間に処理される取引

に適切な業務規則が適用され得るコンピュータシステムを提供することが望ましい。

【0009】

【課題を解決するための手段】前記課題を解決し目的を達成するために、本発明は以下に示す手段を用いている。

【0010】本発明の一つの態様によると、プログラムコードの入力、または修正も要らず、また再コンパイルの必要もなく、再利用プログラム部品からカスタマイズされたACCOUNTSシステムを構築するための開発ツールが提供される。オブジェクト指向プログラミング言語で実装されているので、本発明によるシステムは、ユーザ、またはコンピュータによるプログラムコードの生成、または修正をせずに、また再コンパイルをせずに、プログラム部品のプログラムへの挿入、およびそれからの削除を可能にする。これは継承と多相等のオブジェクト指向プログラミング技術を使うことにより実現される。

【0011】本発明の別の態様によると、一つもプログラムコードを入力、修正、または再コンパイルせずに、既存の再利用部品からユーザ定義の再利用部品を作成可能である。これにより、プログラムコードの再利用が促進され、カスタマイズされたACCOUNTSシステムのもジュール設計が容易になる。

【0012】本発明の更に別の態様によれば、業務規則はオブジェクトとして表現され、業務規則の変化は時間経過に合わせて管理される。これにより、ユーザはある業務規則が有効となる期間を特定でき、そのシステムは、特定の取引が処理される際に適切な業務規則を適用可能になる。

【0013】本発明の更に別の態様によれば、ユーザとシステムとの対話を可能にし、カスタマイズされたACCOUNTSシステムの設計とその設計されたシステムを使用して入力データを処理することを含む全てのタスクを、プログラムコードを一つも入力せずに実現するグラフィック・ユーザ・インターフェイス（GUI）が提供される。例えば、GUIにより、ユーザは、会計システムの部品を表すアイコンを画面上に配置し、そのアイコンを結合する線（部品間の関係を表す）を引くことにより、ACCOUNTSシステムを構築することが出来る。また、GUIにより、ユーザは既存の再利用部品から新たな再利用部品を作成でき、かつ業務規則の変化を定義することが出来る。

【0014】

【発明の実施の形態】以下、図面を参照して本発明によるコンピュータシステムの実施形態を説明する。ここでは、会計システムに適用したコンピュータシステムを例に取り、説明するが、本発明は会計システムに限定されず、種々の分野に適用可能である。

【0015】第1実施形態

本発明の第1の実施形態によるACCOUNTSシステムはコンピュータ上に実装されており、本システムの各機能もコンピュータ上に実装されており、本システムの各機能はこのコンピュータをプログラム形式で記憶されている手順に従い動作させることにより実現される。従って、以下では夫々対応した機能を有する仮想回路ブロック（手段）を想定してこのシステムを説明する。

【0016】コンピュータ・ハードウェア構成は柔軟性がある。一般に、コンピュータはCPU（中央処理装置）とRAM（ランダムアクセスメモリ）から成る主記憶装置により構成されている。コンピュータはどんなサイズでもかまわない。即ち、マイクロコンピュータ、パソコン、小型コンピュータ、ワークステーション、またはメインフレーム・コンピュータが使用可能である。一般に、コンピュータのハードウェアは、必要となるハード・ディスク等の外部記憶装置、およびキーボードとマウス等の入力装置と、CRTディスプレイとプリンタ等の出力装置とI/O制御部を含むユーザ・インターフェイスから構成される。外部記憶装置、入力装置、出力装置として、他の装置の利用も可能である。

【0017】典型的には、コンピュータ上に本システムを実装させるソフトウェアはこのシステムの各機能を実現するアプリケーションプログラムと、その制御の下でこのアプリケーションプログラムが実行されるオペレーティングシステムとから構成される。コンピュータ・ソフトウェア構成は同様に柔軟である。即ち、本発明が実装される限り、ソフトウェア構成は変更しても良い。

【0018】本明細書で使用されている用語「手段（あるいは部分）」はこのシステムの各機能に対応する概念的手段を指している。手段とハードウェア、またはソフトウェアの単位間には必ずしも一対一の対応関係は無い。例えば、同一のハードウェア、またはソフトウェア、部品は時には異なる手段を構成している。

【0019】入力装置を使ってウィンドウを開けたり、ポップアップ・メニューを起動したり、メニューから項目を選択したり、アイコンの位置を定めたり、または幾何学的図形を描いたり等の本明細書に記載されているグラフィック・ユーザ・インターフェイスの特徴を実施する方法は当該技術分野では周知であるので、ここではその記述はしない。さらに、メニュー、ウィンドウ、ボタン、アイコン、線等の実施形態の説明中で種々の図形的特徴を示す用語は、他の同様なグラフィカル表現も含む。

【0020】このシステムはコンピュータ上に実装されるが、その全部、または一部は電子回路に実装されても良い。

【0021】フレームワーク

本発明の実施形態は会計分野について記載されているが、本発明はデータベース、グラフィック・ユーザ・インターフェイス、オペレーティングシステム、在庫管

理、物流、人的資源の管理等の幅広い分野に適用可能である。

【0022】以下にACCOUNTSシステムのフレームワークについて説明する。図1は商店を表す勘定ツリーの実例を示している。最上位レベルの「商店」勘定11には「在庫」勘定12と「買掛金」勘定13が含まれているとする。勘定12と13は勘定11の「要素（部品）」勘定と呼ばれる。「在庫」勘定12は、夫々「りんご」14や「みかん」15等の商品の種類ごとに勘定を持っているとする。「買掛金」勘定13は、「ABC社」勘定16、「XYZ社」勘定17等の仕入先ごとの勘定を持っているとする。勘定14～17等の最下位レベルの勘定は基本勘定であり、そこへ記載された取引を格納する。「在庫」勘定12、および「買掛金」勘定13等のように比較的同質の要素勘定を下位に持つものは分類と呼ばれる。分類勘定の要素勘定は典型的には共通の属性名集合を持っている。「商店」勘定11等の異質の要素勘定を下位に持つものは分割勘定と呼ばれる。

【0023】図2は各種の取引と勘定間の関係だけでなく、図1に示す会計システムが持ち得る取引（二重枠のボックス）の例を示す。例えば、「販売」取引21は「在庫」勘定23のみに影響を与え、「納品書」取引22は「在庫」勘定23と「買掛金」勘定24の両方に影響する。

【0024】取引が転記されると、勘定ツリーのルート勘定（例えば、図1の「商店」勘定）から始まり格納されるべき一つ以上の勘定へ到達するまでそのツリーの下位レベルの勘定へと勘定間で取引が受け渡されていく。異なる種類の勘定はそこへ転記された取引を異なる方法で処理する。基本勘定は単にその取引を格納する。分類勘定は一般的にその要素勘定の一つへその取引を送る。例えば、取引の「商品タイプ」のフィールドの値に応じて、図1の「在庫」勘定は受け取った「販売」取引を「りんご」勘定、または「みかん」勘定のどちらかに送ることになる。この機能は分類機能と呼ばれる。分割勘定は一般に取引を一個以上の新たな取引に変換し、その新取引をその要素勘定の中の一つ以上の勘定へ転記する。システム内で発生された新取引は内部取引と称されることもあり、またユーザにより外部からシステムへ転記された取引は外部取引と称されることもある。例えば、「納品書」取引（外部取引）が「商店」勘定に転記されると、「商店」勘定は二つの内部取引を作成する。その内部取引の一つは売れた商品の量に関する情報を含んでおり、「在庫」勘定に転記される。他方の内部取引は金額に関する情報を含んでおり、「買掛金」勘定に転記される。図3は、「販売」取引31がどのようにして「買掛金」勘定に転記される取引32と「在庫」勘定に転記される取引33に分割されるかを示している。図3に示す取引のデータフィールドの一個以上の内部取引へのマッピングは取引マッピングと呼ばれることもある。

【0025】したがって、会計システムでの各勘定は取引に適用される処理と考えても良い。例えば、基本勘定は取引を格納する処理と考えられる。分類勘定は、取引が分類され、要素処理（要素勘定）の一つへ転記される処理と考えられる。分割勘定は、一つの取引を二個以上の新たな取引に変換し、二つ以上の要素処理（要素勘定）に転記する処理と考えても良い。本発明によれば、これらの処理は別個の再利用プログラム部品としてオブジェクト指向プログラミング・システムに実装されている。

【0026】従って、本明細書で使用されているように、会計フレームワークでの勘定は時には処理、再利用（プログラム）部品、または単に部品とも称される。例えば、上述の処理は夫々「基本」、「分類」、および「分割」部品と称しても良い。

【0027】会計システムのフレームワークの説明をした所で、本発明の各種の形態を以下に説明する。

【0028】図4～図8は会計システム用のグラフィック・ユーザ・インターフェイス（GUI）を勘定ツリーの設計がされ、プログラムが再利用プログラム部品から作成される場合を例にとり示している。図4～図8に示すように、ユーザはコンピュータ画面上でアイコンと線を描くことにより勘定ツリーを設計する（図4～図8の境界は画面上のユーザ・インターフェイス・ウィンドウを表している）。アイコンは再利用プログラム部品を表し、そのアイコンをつなぐ線は一連の処理を表している。コンピュータは、再利用プログラム部品を使い、その勘定ツリーグラフに従ってプログラムを作成する。そしてこのプログラムは取引を処理するために実行される。

【0029】図4に戻って、ユーザはまず入力装置を用いて、利用可能な再利用部品のリストが含まれるポップアップ・メニュー41等の画面表示を起動する。次に、ユーザはメニュー項目42を選択し、表示されるアイコンのために画面上の位置を指示する。選択した再利用部品の図形（アイコン）43aは画面上のユーザが選んだ位置に表示される（図5）。上述の工程は全ての部品43a～43fが画面上の希望位置に置かれるまで繰り返される（図5）。次に、ユーザは入力装置を用い処理ツリーの設計に従って、矢印を画面上に引き部品43a～43fを結び付ける（図6）。図6は引かれた矢印線44と現在引かれている線45と共に線45が引かれているときのカーソル46の位置を示す。矢印線44は連続する処理を示し、矢印線44に対応する矢印線44aは取引が一つの処理から別の処理へ送られる際の方角を表している。

【0030】処理ツリーの中のアイコンの一つは入力部品を示す入力アイコン47である（図7）。図示の例では、入力部品47は分割部品43aに接続される。入力部品は処理ツリーで定義される一連の処理の始点を表

し、取引を入力するために使用される。ユーザは入力定義ウィンドウ 4 8 を起動する（例えば、入力アイコンの位置でマウスカーソルをクリックすることにより起動する）。これにより、ユーザは取引が持つことになるデータ・フィールド等の入力フォーマットを定義できる。入力定義ウィンドウ 4 8 はこの入力フォーマット内のデータ・フィールドを追加、削除できるようにポップアップ・メニュー 4 8 a を含む。

【0031】この入力フォーマットを定義後、ユーザは入力ウィンドウ 4 9 を起動し取引用のデータを入力する（例えば、入力アイコンの位置でマウスカーソルをクリックすることにより起動する：図 8）。全てのデータ項目の入力が終了後、ユーザは入力ウィンドウ 4 9 の了解ボタン 4 9 a をクリックする（図 8）。これにより、図 9 に示すように、ユーザが描いた処理ツリーが表す連続処理に従って進行する入力データの処理が始まる（図 9 において、括弧で括られているテキストとそのテキストの隣にある矢印は G U I 表示の一部ではなく、単に入力データが各部品によりどのように処理されて、渡されるかを説明するために入れてある）。ユーザが定義した処理ツリーはコンピュータに記憶しておいて、後で入力データを処理するために再表示、および使用される。

【0032】図 4～図 7 に示した例では、部品アイコンを配置してから矢印線を引いたが、アイコン（入力アイコンも含む）と矢印線はどのような順序で画面上に描いても良い。例えば、いくつかのアイコンを結ぶ矢印線は他のアイコンを画面上に描くよりも早く描いても良い。さらに、矢印線が部品間の関係を、つまりアイコンによって表される連続処理を概念レベルで示す限り、部品アイコンの実際の配置位置とその線は重要ではない。

【0033】G U I のアイコンは再利用プログラム部品、即ち処理単位を表している。再利用部品は、メソッドをカプセル化するオブジェクト、すなわちそのメソッドが属するクラスとしてオブジェクト指向プログラミングシステムに実装され得る既存のプログラム部品である。G U I を使用することにより、ユーザは、プログラムコードを一つも生成、変更、または再コンパイルせずに既存のプログラム部品からプログラムを作成する。この処理は「プログラム組み立て」と呼ばれ、以下に記述するオブジェクト指向技術を用いて達成される（別段の定めが無い限り、以下の説明で使用される例題は C++ 文法で書かれている）。

【0034】プログラム組み立ての実装  
非オブジェクト指向プログラミングシステムでは、再利用プログラム部品は一般に関数の形式で提供されている。その様なプログラム部品を挿入したり削除したりするには、プログラムのコンパイルが必要となる。他方、オブジェクト指向プログラミングシステムでは、再利用部品はオブジェクトの形で提供される。オブジェクトは属性やメソッドが定義されるクラスから生成される。本

明細書では、クラスで定義されるメソッドを、クラスまたはクラスから作成されるオブジェクトのメンバー関数と呼ぶこともある。あるオブジェクトのあるメンバー関数はその関数を参照する、すなわちメッセージをそのオブジェクトに渡すことにより起動可能である。オブジェクト指向プログラミングシステムでは、オブジェクトはそのメンバー関数のコンパイルされたコードに対するポインタの集合を得る。オブジェクトの関数が起動されると、その関数を起動するプログラムコードはその起動された関数へのポインタに置き換えられる。これは、プログラムのコンパイル中ではなくその実行中に、動的に実施される。

【0035】プログラム組み立てをコードを生成または再コンパイルせずに達成するために、全ての再利用・プログラムは同一のインターフェイスを有するクラスの形で提供される。

【0036】

```
C1::f(arg1, ..., argN)
C2::f(arg1, ..., argN)
C3::f(arg1, ..., argN)
C4::f(arg1, ..., argN)
```

...

ここで、「C1::f」は、クラス「C1」のメソッド「f」を参照する C++ 文法である。上述の例では、各クラス C1, C2, C3, C4, ... は同一インターフェイス f(arg1, ..., argN) を有する関数 f を実装する。多相性の原理により、それらの関数はそれが定義されるクラスによって異なる演算を表す。例えば、関数 C1::f, C2::f, C3::f と C4::f は夫々基本処理（入力データの格納）、分類処理（入力データの分類）、分割処理（入力データの分割）と変換処理（入力データの変換）を実装しても良い。しかし、これら関数の全ては同一インターフェイス f(arg1, ..., argN) を有している。

【0037】従って、本発明の一つの形態では、全ての再利用プログラム部品は同一インターフェイスを持っている。これは、スーパークラスから得られたサブクラスがそのスーパークラスと同一のインターフェイスを有することになる継承機構を使ってオブジェクト指向プログラミングシステム内で達成される。好適には、一つのスーパークラスは、会計システムの全ての再利用部品用の共通インターフェイスを定義するために使用される。

【0038】プログラム組み立てを達成するために、配列、リスト、または辞書等のデータ構造は任意の部品の集合を保持し、後続の部品の挿入、および削除を容易にするために使用される。その様なデータ構造を使用することにより、その部品の連続起動もまた容易になる。例えば、そのソフトウェアのメイン関数は下記のように実装されても良い。

【0039】

```

main () {
    ...
    for (i=0, i<n, i++) {
        a[i].f(arg1, ..., argN);
    }
    ...
}

```

ここで  $a[i].f$  はオブジェクト  $a[i]$  中のメソッド  $f$  を起動する C++ 文法である。上述の例では、配列  $a[i]$  はオブジェクトの集合を保持するのに使用される。そのオブジェクト  $a[i]$  のメンバー関数  $f(arg1, \dots, argN)$  はループを使用して起動され、それぞれのメンバー関数により定義された操作が順番に実行される。再利用部品を挿入（または削除）するために、所望の再利用部品（オブジェクトとして実装されている）は配列要素  $a[i]$  に割当てられる。例えば、上記の例では、入力データを分類しそのデータを基本勘定に格納する処理は、クラス C2（これは分類処理を実装する）に属するオブジェクトを配列要素  $a[0]$  に割当て、クラス C1（これは基本処理を実装する）に属するオブジェクトを配列要素  $a[1]$  に割当てることにより実装される：

```

a[0] = new C2;
a[1] = new C1;

```

従って、このループが実行されると、分類部品、および基本部品は対応する処理を実装するために実行される。

```

SuperClass a[N];
Class B::f(arg1, ..., argN)
{
    ...
    for (i=0, i<N, i++) {
        a[i].f(arg1', ..., argN');
    }
    ...
}

```

ここで、「クラス B」はオブジェクト B が属するクラスであり、「スーパークラス」は好適にはクラス B が継承しているスーパークラスである。上述の例では、後続部品は配列  $a[N]$  に保持される。後続部品は起動する部品が継承するスーパークラスと同一のクラスから継承するのが好適である。好適には、関数インターフェイス  $f(arg1, \dots, argN)$  は元来そのスーパークラスで宣言されている。

【0042】オブジェクトの配列要素に対する割当てはユーザが GUI を介してラン・タイムで実行される。例えば、部品 B と C を表すアイコンが表示画面上に置かれ、部品 B から部品 C に線が引かれると、部品 B は部品 C を配列要素に割当てて、このような割当てにはプログラムコードの変化は全然無く、従って再コンパイルは必要が無い。従って、上述のオブジェクト指向プログラミ

新しい部品の挿入は挿入された部品を適切な配列要素に割当てることにより行われる。例えば、以下の割当てにより、変換処理（クラス C4 に属するオブジェクトにより実装されている）が前列での分類部品（クラス C2 に属するオブジェクトにより実装されている）と基本部品（クラス C1 に属するオブジェクトにより実装されている）間に挿入されることになる。

```

【0040】 a[0] = new C2;
a[1] = new C4;
a[2] = new C1;

```

このシステムは一連の起動を可能にする。例えば、部品 A によって起動された部品 B が部品 C を更に起動することが可能となる（本明細書では、部品が起動されると記述するが、オブジェクト指向プログラミング言語の文脈では、起動されるものは一般にオブジェクトのメソッドである）。これは、後続の部品（例えば、部品 A に対する部品 B）の各メンバー関数を、次のように実装することにより実現される。すなわち、部品（部品 B）のあるメンバー関数が先行する部品（部品 A）によって起動されるとき、後続部品（部品 B）は、それ自身の（B の）後続部品（例えば、部品 C）の同一名の関数を起動する。例えば、部品 B のメンバー関数  $f(arg1, \dots, argN)$  は下記のように実装される。

```

【0041】

```

ング技術を利用することによりプログラム組み立てを達成出来る。これにより、再利用部品がユーザ、またはコンピュータによるプログラムコードの生成、または修正無しに、かつ再コンパイル無しにプログラムを組み立てるのに使用される。

【0043】ここで、図 9 を参照して、各部品のメンバー関数の起動処理を説明する。まず、図 9 では、「分割」部品から「変換」部品、「分類」部品にそれぞれ矢印線が引かれ、「変換」部品から「基本（税金）」部品に矢印線が引かれ、「分類」部品から「基本（XYZ 社）」部品、「基本（ABC 社）」部品に矢印線が引かれている。

【0044】そのため、「分割」部品内では、「変換」部品、「分類」部品が配列要素に割り当てられる。例えば、 $a[1]$  = 「変換」部品、 $a[2]$  = 「分類」部品とい

うようにである。また、「変換」部品内では、「基本(税金)」部品が配列要素に割り当てられており、a[1] = 「基本(税金)」部品となっている。さらに、「分類」部品内では、「基本(XYZ社)」部品、「基本(ABC社)」部品が配列要素に割り当てられ、a[1] = 「基本(XYZ社)」部品、a[2] = 「基本(ABC社)」部品となっている。

【0045】次に、処理手順を説明する。図9で、「入力」部品から「分割」部品が起動されると、「分割」部品のメンバー関数が実行される。すると、配列要素としてa[1] = 「変換」部品、a[2] = 「分類」部品となっているので、まず、「変換」部品が起動される。

【0046】「変換」部品が起動されると、さらに「変換」部品のメンバー関数が実行され、配列要素として割り当てられている「基本(税金)」部品が起動する。

【0047】そして、「変換」部品の実行が終了すると、「分類」部品が起動される。「分類」部品が起動されると、「分類」部品のメンバー関数が実行され、a[1]に割り当てられている「基本(XYZ社)」部品が起動される。「基本(XYZ社)」の処理が終了すると、a[2]に割り当てられている「基本(ABC社)」部品が起動される。

【0048】このように、「分類」部品→「変換」部品→「基本(税金)」部品→「分類」部品→「基本(XYZ社)」部品→「基本(ABC社)」部品というように、一連の起動が可能となるものである。

#### 【0049】システム構成

本発明の一実施形態の機能ブロック図である図10を参照してこの会計システムの構成を説明する。図中の各機能ブロックは別個のハードウェア、またはソフトウェア部品である必要はないが、ハードウェア、またはソフトウェアによって実現可能である。

【0050】図10に示すように、このシステムはキーボード、および/またはマウスであっても良い入力装置101と、CRTであっても良い表示ユニット102を含む。入出力制御部(I/O制御部)103は入力装置101と、表示ユニット102とその他のシステムの部分間のインターフェースを管理する。このシステムには更に、再利用部品の定義、コンパイル、組み立て、および実行を全てシステムのバックグラウンドで管理するオブジェクト指向プログラミング及び実行環境部104が含まれている。再利用部品表示部106はオブジェクト指向プログラミング及び実行環境部104からの再利用部品の定義を受け、図4に示すポップアップメニューのようにユーザのブラウジングを容易にするフォーマットでの定義を表示する。再利用部品定義部105はユーザが再利用部品の定義を行えるようにし、ユーザ定義の部品がその他の再利用部品と協調するように新しい再利用部品を定義する際に適切な制約を課す。また、定義部105はユーザ定義の再利用部品の定義をオブジェクト指

向プログラミング及び実行環境部104に登録する。

【0051】再利用部品選択部107は、入力装置101からの選択入力(ユーザ選択の再利用部品を表す)に応答し、複合表示部108に指示して、ユーザが選択した再利用部品の図形を表示ユニット102に表示させる。選択入力は、例えば、再利用部品表示部106で表示されたメニューから選択されたものである。その再利用部品の図形は、例えば、図10に示すアイコン43a~43fである。再利用部品接続部110は、再利用部品間でユーザが定義する接続を表す入力装置101からの接続入力に応答して、複合表示部108に、ユーザが定義した接続の図形を表示ユニット102に表示させるように指示する。接続入力とは、例えば、ある再利用部品を表すアイコンを別の部品を表す別のアイコンと結びマウス操作である。そのユーザ定義された接続の図形は、例えば、図11に示す矢印線44のように、アイコンを連結する矢印の形である。

【0052】再利用部品選択部107と再利用部品接続部110からの要求で再利用部品、およびそれら間の接続の図形を表示することに加えて、複合表示部108は受け取った選択、および接続入力を表すデータを複合データファイル109に記憶させる。この選択、および接続入力データに従って、複合表示部108はオブジェクトと、このオブジェクトの一連の起動関係をオブジェクト指向プログラミング及び実行環境部104で作成する。従って、表示部108は再利用部品から組み立てられたプログラムの再生成を容易にする。

【0053】入力データフォーマット定義部111は入力装置101の要求に応答し、ユーザが入力フォーマットを定義できるように複合表示部108に指示を出してデータと図式表現を表示ユニット102に表示させる。このデータと図式表現は、例えば、図7に示すようにウィンドウ48の形で表示される。また、この入力データフォーマット定義部111はユーザにより定義される入力フォーマットを表す入力装置101からの入力を受け取り、定義された入力データフォーマットを入力データフォーマットファイル112に記憶させる。データ入力部113は入力装置101からの要求に応答し、複合表示部108に指示してそのデータと図式表現を表示ユニット102に表示させて、ユーザが入力データフォーマットファイル112に記憶された入力データフォーマットに応じてデータを入力できるようにする。このデータと図式表現は、例えば、図8に示すようにウィンドウ49のような形で表示される。データ入力部113は更にユーザが入力したデータ、即ちACCOUNTSシステムで処理される入力データを表す、入力装置101からの入力を受け取る。

【0054】処理開始部114は、データ入力部113からの入力データを受け取り、この入力データの処理に関するユーザからの要求を表す、入力装置101からの

要求にตอบสนองして、作成されたプログラム（つまり、対応するオブジェクトの組み立て）をオブジェクト指向プログラミング及び実行環境部 1 0 4 で実行する。再利用部品状態表示部 1 1 5 はオブジェクト指向プログラミング及び実行環境部 1 0 4 でのオブジェクトの状態を表示ユニット 1 0 2 に表示する。この表示により、ユーザはデータ処理結果を見ることが容易となる。

#### 【0055】カスタム再利用部品の定義

図 9 に示す「分割」、「分類」、「変換」部品等の各再利用部品により実行される特定の関数は G U I により定義される。例えば、分割部品は転記されてきた取引を分割し、取引マッピングに応じて、異なる低位レベルの勘定のデータフィールドに渡す。ユーザは G U I を通して取引マッピングを指定することによりカスタム分割部品を定義する。分類部品は特定のフィールドの値に応じて取引を複数の低位レベルの勘定の一つに渡す。ユーザは取引が渡された低位レベルの勘定と特定のフィールドの値の対応関係を G U I を通して指定することによりカスタム分類部品を定義する。

【0056】しかしながら、変換部品の処理は複雑であるので、カスタム変換部品の定義は複雑である。図 1 1 には取引 6 1 が変換部品「給与」6 2 により処理される方法が示されている。この例では、変換部品「給与」6 2 は毎月の給与と残業手当データを含む取引 6 1 を受け取ると、変換部品「給与」6 2 は給与データを含む取引を給与勘定 6 3 に転記し、入力された残業データから残業手当を計算し、それを残業手当勘定 6 4 に転記し、給与と残業手当を加算して給与総額を計算し、それを給与総額勘定 6 5 に転記し、税金データを計算して、それを税金勘定 6 6 に転記し、税金をその給与総額から減算し、手取りを計算しそれを、預金勘定 6 7 に転記する。この例における下位レベル勘定 6 3 ~ 6 7 の全ては、そこに転記された取引を格納する基本勘定である。変換部品「給与」はユーザが G U I を以下のようにして使用し定義することが可能である。

【0057】このような変換部品は本明細書で説明したプログラム組み立て技術を用いて、プログラムコードを入力、修正、または再コンパイルせずに、1 つ、または複数の再利用部品から構成される。

【0058】図 1 1 に示す給与部品の機能を実現するための変換部品を構成する際の G U I を図 1 2 ~ 図 1 4 に示す。図 1 2 に示すように、（未定義の）変換部品を含む処理ツリー 7 1 が設計され、先ず画面に表示される。ユーザは例えば「変換」アイコン 7 3 に位置するマウスカーソル 7 2 をクリックすることにより、変換部品を定義する。これに応じて、表示画面には定義される変換部品と協働する部品（ここでは、取引を変換部品に転記する入力部品 7 4 と、変換部品が転記される基本部品 7 5 a ~ 7 5 e）と、定義される変換部品を定義するための空白エリアが表示される（図 1 3）。図 4 ~ 図 6 を参照

して前述した入力技術を用いて、ユーザはメニュー（図示せず）から既存の再利用部品、例えば給与、残業手当、給与総額、税金、預金等を選択し、その選択された部品に対応するアイコン（例えば、7 6 a ~ 7 6 e）を画面上に配置する（図 1 4）。これらの再利用部品は、変換部品「給与」の操作を定義するために使用される規則、公式等を表わすので、変換規則と呼称される。変換規則部品はシステムに設けられていても良いし、オブジェクト指向プログラミング及び実行環境部 1 0 4 を用いてプログラマーにより予め作成されていても良い。一度作成されると、変換規則部品は他の再利用部品と同様に扱われる。

【0059】ユーザは種々の部品を接続する矢印線（例えば、実線 7 7 a ~ 7 7 j、点線 7 8 a ~ 7 8 b）を引く。実線と点線は、例えばマウスの左ボタンを押しながらカーソルをドラッグする、あるいはマウスの右ボタンを押しながらカーソルをドラッグする等の異なる入力技術を用いて引いてもよい。各実線は宛先の部品の起動を示す。起動には、起動側部品から宛先部品への取引の転記も含まれる。例えば、実線 7 7 a は入力部品 7 4（起動側部品）により給与部品 7 6 a（宛先部品）が起動され、入力された取引が給与部品へ転記されることを示す。各点線は宛先部品によるソース部品の属性の参照を示す。例えば、点線 7 8 a は給与総額 7 6 c（宛先部品）による給与部品 7 6 a（ソース部品）の属性（給与データ）参照を示す。宛先部品によるソース部品の属性の参照はソース部品の機能（例えば、参照"referTo"機能）を宛先部品から起動することにより実現される。この機能は、ソース部品の参照される属性を宛先部品に返す。

【0060】図 1 4 の例では、実線 7 7 c に示すように入力部品 7 4 により給与総額部品 7 6 c が起動されると、それぞれ点線 7 8 a、7 8 b に示すように給与部品 7 6 a と残業手当部品 7 6 c の参照が開始される。給与総額部品 7 6 c は給与部品、残業手当部品から返されたデータを用いて給与総額データを計算する。この給与総額データは給与総額部品の属性となり、続いて、例えば税金部品 7 6 d、預金部品 7 6 e により参照される。

【0061】一度処理ツリーが定義されると（図 1 4）、システムは再利用部品 7 6 a ~ 7 6 e と起動に基づいて変換部品を作成し、実線 7 7 a ~ 7 7 g、点線 7 8 a ~ 7 8 e で定義されるように参照を行う。このようにユーザにより定義された給与部品は入力データ（ここでは入力部品により転記された取引）を処理するために使われ、後で利用できるように保存される。

【0062】既存の再利用部品から再利用部品を作成することは前述した同一のオブジェクト指向プログラミング技術で実装される。特に、後続の部品の同一名のメンバー関数を起動させるある部品のあるメンバー関数（先行部品により起動された）の能力が利用される。この技

術により一連の処理の構築が容易となる。

【0063】上述の例は、ユーザによる変換部品の定義を示した。一般的に、ユーザは既存の再利用部品を用いて所望の機能を実現する新しい再利用部品を構成する。ユーザが定義した再利用部品はデータ処理のための処理ツリーを構築するため、あるいは新しい再利用部品を定義するために使われる。これは、コードの再利用を促進し、モジュール化、あるいは階層化されたシステム設計を容易にする。このため、ユーザは大規模な、そして複雑なプログラムモジュールを簡単に設計することができる。上記の動作はプログラムコードを入力、修正、または再コンパイルせずに、行われる。

#### 【0064】業務規則の変化

このACCOUNTSシステムがサポートする別の特徴は業務規則の変化を追跡することである。本発明のオブジェクト指向プログラミング技術を用いると、業務規則の変化を時間を追って追跡することができる。これにより、過去に発生した取引の処理が可能になり、取引が発生した時点で有効な業務規則に従った取引の処理が行われることが確保される。

【0065】業務規則の変化は部品間の接続が有効となる期間に関して定義される。業務規則の変化の一例としては、業務組織の組織表の一部を示す図15に示されているものがある。図15に示すように、業務単位の「オブジェクト指向技術グループ」81は、1997年9月30日以前は業務単位の「R&Dセンター」82のサブユニットであり、1997年10月1日に「ソフトウェア業務ユニット」83のサブユニットになっている。従って、部品82と部品81間のリンク84aは1997年9月30日より前の期間でのみ有効であり、部品83と部品81間のリンク84bは1997年10月1日以降の期間でのみ有効である。図15の下位レベルの部品85はユニット81のサブユニットを表し、部品81と部品85間のリンクは両期間で有効のままである。

【0066】図16～図19には変化する業務規則を定義し、適用するためのグラフィック・ユーザ・インターフェイスの利用が図示されている。処理ツリー91が表示画面に表示されると、ユーザは特定のリンク、即ち、部品93aと93b間のリンク92の有効期間を、そのリンク92を指示しているマウスカーソル94をクリックすることにより定義可能である(図16)。それに応答して、ウィンドウ95のようなプロンプトが画面に表示され、ユーザはリンク92が有効となる期間を入力できる(図17)。このシステムはユーザによって入力された全ての業務規則用の有効な期間を記憶し、それにより業務規則の変化を時間を追って追跡できる。作成されたプログラムが実行され取引が処理される際に、ユーザはウィンドウ96のような画面表示により処理される取引の発生日(モデル日付と称される)を入力するように促される(図18)。このモデル日付は任意の日付であ

る。プログラムは次にそのモデル日付に有効な業務規則に従って取引を処理する。加えて、そのプログラムが特定のモデル日付に実行された場合は、処理ツリーはそのモデル日付に有効な業務規則に従って画面に表示される。例えば、そのプログラムが部品93aと93b間のリンク92の有効期間外のモデル日付に実行されると、そのリンク92は処理ツリーグラフから消えてしまう(図19)。

【0067】業務規則の変化を追跡する処理は、部品間の接続有効期間をオブジェクトとして表すことによりオブジェクト指向プログラミングで実装される。たとえば、図20の(a)と(b)に示すように、ユーザが部品Aと部品B間のリンク201の有効期間を(GUIを介して)定義すると、有効期間部品202が部品Aから部品Bへの一連の起動関係(a)に挿入される(b)。あるいは、有効期間部品202はユーザが初めてリンク201を生成した時に自動的に挿入されてもよい。後者の場合、リンク201が有効である期間はユーザにより定義されない限り無限である。

【0068】上述したオブジェクト指向プログラミング技術は、有効期間部品202の挿入と、プログラムコードの入力、または修正あるいはそのコードの再コンパイル無しに、図20(b)に示す一連の起動関係を作る際に使用される。特に、有効期間部品202は部品Bと同じインターフェイスを持つように実装され、これにより部品Aは部品202中のメンバー関数を部品B中の同名のメンバー関数を起動させるのと同じ方法で起動することが出来る。有効期間部品202中のメンバー関数が部品Aにより起動されると、それはモデル日付とそこに含まれる有効期間を比較する。モデル日付がこの有効期間内であれば、有効期間部品202中のメンバー関数は部品B(後続部品)中の同名のメンバー関数を起動する。一方、モデル日付がこの有効期間外であれば、有効期間部品202中のメンバー関数は部品B中の同名のメンバー関数を起動せず、一連の起動は終了する。

【0069】業務規則の変化の別の例は、ある部品が二個以上の後続部品を持ち、ある有効期間内に一個の部品を選択的に起動する場合である。この二個以上の後続部品の有効期間は相互排他的である。即ち、これらの有効期間は夫々重なることはない。この状態の一例としては、業務組織表の一部を示す図21に示されているものがある。この例では、「R&Dセンター」は1997年9月30日以前は「オブジェクト指向技術グループ」を含んでいるが、1997年10月1日からは、オブジェクト指向技術はアウトソーシングにより取り扱われる。図12に示すように、この状況での業務規則の変化の追跡は、起動部品(「R&Dセンター」221)内のディクショナリ等のデータ構造を使い、かつその起動部品221と二個の有効期間部品223aと223b間に「部品・セット」オブジェクト222を挿入することにより

オブジェクト指向プログラミングで実装される。

【0070】ディクショナリとセットはオブジェクト指向プログラミングではコレクションクラスの範疇に入る。コレクションは一つのオブジェクトであり、これには他のオブジェクトのグループが含まれる。各ディクショナリは複数対のオブジェクトのリストであり、各対は「キー」と「値」から成る。一つのディクショナリ内のこれらのキーと値はオブジェクトである。一方、セットは複製がないコレクションである。

【0071】図22に実質的に示されるように、起動部品であるR&Dセンター221はディクショナリ224を使用する。このディクショナリ224では、キー「オブジェクト指向技術グループ」226の値225は部品・セット222である。この部品・セット222は相互排他的有効期間を定義する一群の有効期間オブジェクト227a、227b、…を保持する。図20(b)に示す例の場合と同様に、有効期間オブジェクト中のあるメンバー関数を起動する代わりに、本例での起動部品221は部品セットオブジェクト222中の同名のメンバー関数を起動する。次に、部品セットオブジェクト222中のこのメンバー関数は、起動が発生する時間（即ち、モデル日付）によって、有効期間オブジェクト223a、223b、…の内の一つのオブジェクト中の同名のメンバー関数を起動する。

【0072】部品セットオブジェクトと有効期間オブジェクトは部品221と部品223a、223bとの間のリンクが生成される時、あるいはユーザにより相互排他的有効期間が定義される時に、コンピュータにより自動的に挿入される。新しい有効期間オブジェクトを挿入する前に、部品セットオブジェクト222は追加すべき新しいオブジェクトで定義されている有効期間が部品・セット222に保持されている他のオブジェクトの期間と排他的であるかどうか判定する。新しい有効期間オブジェクトは定義されている有効期間が他のオブジェクトの期間と排他的であるか場合のみ挿入される。

【0073】前述のオブジェクト指向プログラミング技術を使うことにより、部品セットオブジェクトの挿入と部品・セット内への有効期間の追加はプログラムコードの入力、または修正あるいはそのコードの再コンパイル無しに実施され得る。

【0074】従って、本発明のオブジェクト指向プログラミング技術を使うことにより、時間経過につれての業務規則の変化は追跡することが可能である。これにより、取引は何時でも処理可能であり、転記された取引はモデル時間に有効な適切な業務規則に従って処理される。

【0075】図23は業務規則の変化の追跡を実施する他の実施形態の構造を示す部分的な機能ブロック図である。図23ではいくつかの要素が図示省略されているが、図10に示す構造の全ての機能的構成要素は本実施

形態に含まれている。図23に示すように、接続管理部116は、二個の部品間の接続の有効期間についてのユーザ定義を表す入力データを入力装置から受け取り、複合表示部108に指示して入力された有効期間に応じて接続関係を示すグラフィック表示を行う。複合表示部108は有効期間入力を表すデータを複合データファイル109に記憶させ、オブジェクトとそのオブジェクトの一連の起動をオブジェクト指向プログラミング及び実行環境部104で作成する。モデル選択部117はモデル時間を表す入力データを入力装置から受け取り、複合表示部108に指示してそのモデル時間に応じて図形を表示する。例えば、表示部108は、もしこのモデル時間が処理ツリー中の二個の部品間の接続の有効期間内であればその接続を表示し、もしこのモデル時間がその有効期間外であれば表示されている処理ツリーからその接続を削除する（図19参照）。加えて、モデル選択部117はグローバル変数等のそのプログラムの適切な変数内にそのモデル時間を設定する。その結果、このモデル時間はオブジェクト指向プログラミング及び実行環境部104で全てのオブジェクトによりアクセス可能となる。

【0076】本明細書で説明した実施形態では、カーソル・ポインティング、マウスボタン・クリック等の特定の入力装置と入力技術を用いてGUIを説明してきたが、タッチパネル、ライトペン、トラックボール等の他の入力装置と入力技術を用いてGUIを構成しても良い。

【0077】このACCOUNTSシステムとユーザ間のインターフェイスはグラフィック・ユーザ・インターフェイス（GUI）として説明されている。ユーザがこのシステムとの対話に要する全ての情報を入力するのには、非グラフィックすなわちテキストによるユーザ・インターフェイスを使用しても良い。そのような非グラフィック・ユーザ・インターフェイスは、例えばテキスト形式のプロンプトでユーザに入力を促す方法を使っても良い。例えば、図4に示すポップアップメニューの代わりに、利用可能な再利用部品のリストをテキスト表示し、ユーザに選択した再利用部品の名前を入力するよう促すプロンプトを表示しても良い。同様に、プロンプトを使用して、例えば接続に対する始点と終点の部品の名前の入力を促すことでユーザに再利用部品間の接続を定義するように要求しても良い。更に、グラフィックと非グラフィックが混合したユーザ・インターフェイスも使用可能である。グラフィック・ユーザ・インターフェイスの場合同様に、非グラフィック・ユーザ・インターフェイスでも、ユーザはプログラムコードを一つも入力する必要は無いし、このシステムによってプログラムコードが生成されることもない。使用するユーザ・インターフェイスの種類に関係なく、本明細書で説明したオブジェクト指向プログラミング技術は、同じくプログラムの変更や再コンパイルせずにプログラムを作成することに

も適用される。更に、図 10 と図 23 に示す本システム構造の機能ユニットを動作させて、使用しているユーザ・インターフェースの種類に応じて適切な表示を作成することも可能である。

【0078】本明細書で説明した、業務規則の時間的な変化を実装するオブジェクト指向プログラミング技術は、ユーザの特徴等の他の条件で変化する業務規則を実装するのに使用できる。ここで述べているオブジェクト指向プログラミング技術により、ユーザはいかなるプログラムコードの生成、変更、または再コンパイルをすることなく、ラン・タイム時にそのような条件を定義することが可能となる。これにより、実現世界のオブジェクトとそのオブジェクト間の関係のモデルを作る際のソフトウェアの柔軟性、および万能性を高めることができる。

【0079】本発明の実施形態と応用例について示し、説明してきたが、本発明の範囲内でその他多くの変更が可能であることは明らかである。従って、ここで開示した実施形態は実例として挙げたものであって、本発明を何ら限定するものではなく、本発明の範囲はこれまでの記載内容からと言うよりは特許請求の範囲によって示されるものである。特許請求の範囲はそのような実施形態と変更例、および本発明の実際の範囲内におけるその他の変更例を保護するものである。従って、特許請求の範囲で意味するもの、および特許請求の範囲から派生する変更は本発明の中に包含されるものである。

【0080】たとえば、本発明が実施できる限り、上述のハードウェア部品の何個かを追加、交換、または削除しても良い。例えば、このシステムは複数のコンピュータが接続されたコンピュータネットワーク上で設置されても良い。CPU はどんなタイプのものでもかまわない。複数の処理を同時に実行するために複数の CPU が使用されても良いし、一個の CPU が時分割で使用されても良い。他の種類の入力装置を使用しても良い。例えば、タッチパネル、ライトペン、またはトラックボール等のポインティング装置や、ディジタイザー、イメージリダー、またはビデオカメラ等の画像入力装置と、音声認識装置と、各種センサを使用しても良い。他の種類の外部記憶装置を使用しても良い。例えば、フロッピーディスクドライブ、RAM カードユニット、磁気テープユニット、光ディスクユニット、光磁気ディスクユニット、バブルメモリユニット、またはフラッシュメモリが使用可能である。他の種類の出力装置を使用しても良い。例えば、液晶表示部、プラズマ表示部、ビデオプロジェクタ、LED 表示部、音響発生回路、または音声合成器を使用してもかまわない。

【0081】

【発明の効果】以上説明したように本発明によれば、プログラムコードを入力、修正、または再コンパイルせずに、新しい、または修正されたプログラムを組み立てる

コンピュータシステムを提供することができる。さらに、グラフィック・ユーザ・インターフェイスを利用して新しい、または修正されたプログラムを組み立てるコンピュータシステムを提供することができる。また、業務規則の変化を時間とともに追跡可能で、所定の時間に処理される取引に適切な業務規則が適用され得るシステムを提供することができる。

【図面の簡単な説明】

【図 1】本発明によるコンピュータシステムを会計システムに適用する場合の勘定ツリーの実例を示す図。

【図 2】図 1 に示す勘定ツリー・システム用の取引例を示す図。

【図 3】引取マッピングの実例を示す図。

【図 4】本発明の実施形態によりグラフィック・ユーザ・インターフェイスを用いるプロセス・ツリーを示す図。

【図 5】本発明の実施形態によりグラフィック・ユーザ・インターフェイスを用いるプロセス・ツリーを示す図。

【図 6】本発明の実施形態によりグラフィック・ユーザ・インターフェイスを用いるプロセス・ツリーを示す図。

【図 7】本発明の実施形態によりグラフィック・ユーザ・インターフェイスを用いるプロセス・ツリーを示す図。

【図 8】本発明の実施形態によりグラフィック・ユーザ・インターフェイスを用いるプロセス・ツリーを示す図。

【図 9】図 4～図 9 に示すプロセス・ツリーによる取引の処理を示す図。

【図 10】本発明の実施形態の構造を示す機能ブロック図。

【図 11】変換部品の実例を示す図。

【図 12】本発明の実施形態によるグラフィック・ユーザ・インターフェイスを使用する、図 11 の変換部品の作成を示す図。

【図 13】本発明の実施形態によるグラフィック・ユーザ・インターフェイスを使用する、図 11 の変換部品の作成を示す図。

【図 14】本発明の実施形態によるグラフィック・ユーザ・インターフェイスを使用する、図 11 の変換部品の作成を示す図。

【図 15】業務規則変更の一例を示す図。

【図 16】グラフィック・ユーザ・インターフェイスを使用した業務規則の変更の有効化を示す図。

【図 17】グラフィック・ユーザ・インターフェイスを使用した業務規則の変更の有効化を示す図。

【図 18】グラフィック・ユーザ・インターフェイスを使用した業務規則の変更の有効化を示す図。

【図 19】グラフィック・ユーザ・インターフェイスを

使用した業務規則の変更の有効化を示す図。

【図 20】オブジェクト指向プログラミング技術を使用して業務規則の変化を追跡する方法を示す図。

【図 21】業務規則を変更する別の例を示す図。

【図 22】オブジェクト指向プログラミング技術を使用して図 21 に示す例の業務規則変化を追跡する方法を示す図。

【図 23】業務規則の変化を実施する本発明の実施形態の構造を示す部分機能ブロック図である。

【符号の説明】

101…入力装置

102…表示ユニット

103…I/O制御部

104…オブジェクト指向プログラミング及び実行環境部

105…再利用部品定義部

106…再利用部品表示部

107…再利用部品選択部

108…複合表示部

109…複合データファイル

110…再利用部品接続部

111…入力データフォーマット定義部

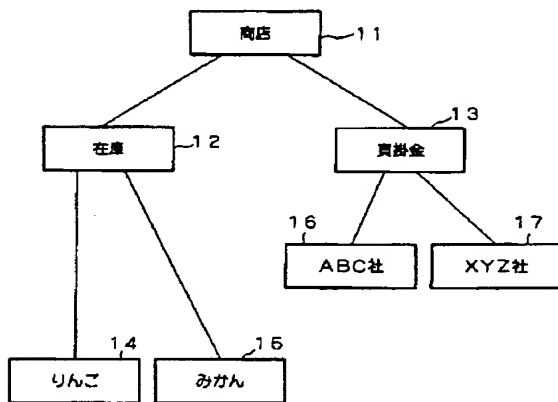
112…入力データフォーマットファイル

113…データ入力部

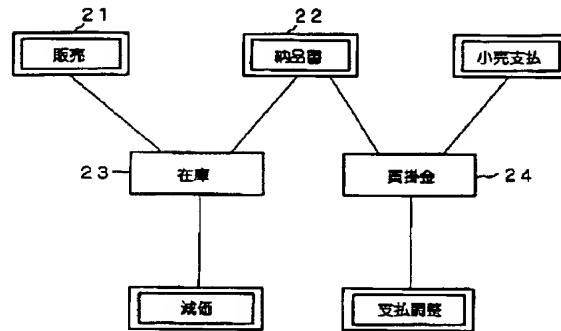
114…処理開始部

115…再利用部品状態表示部

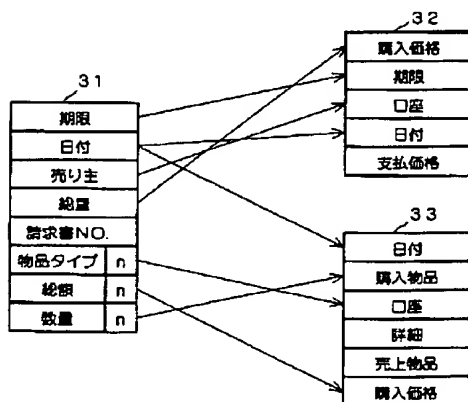
【図 1】



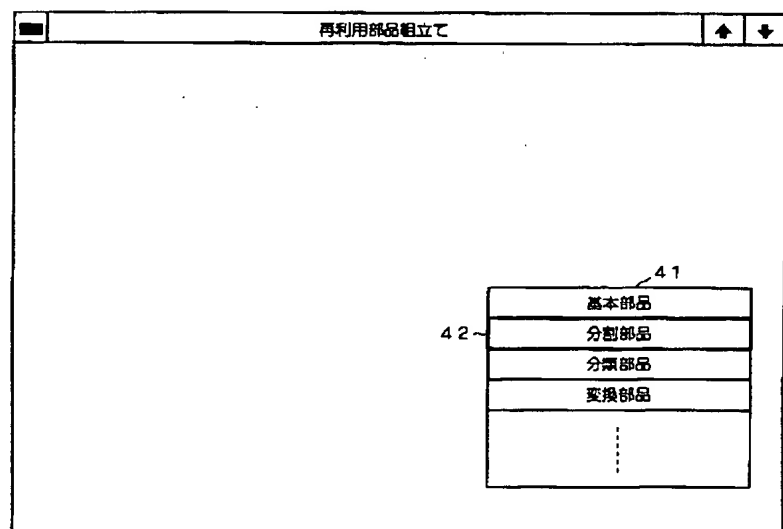
【図 2】



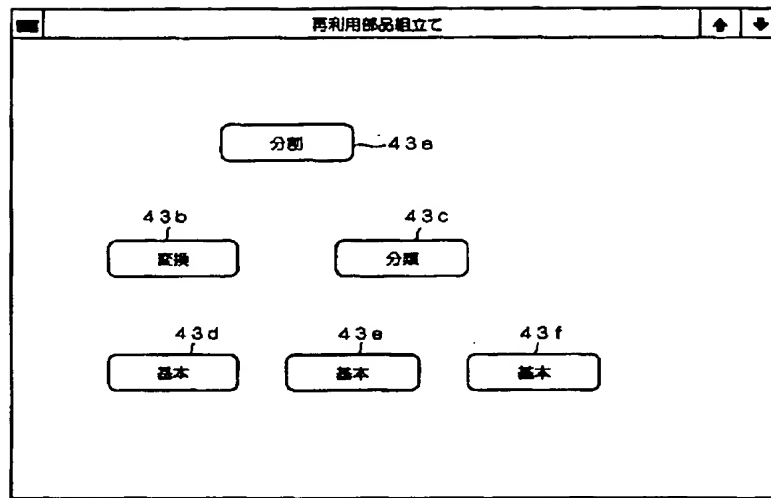
【図 3】



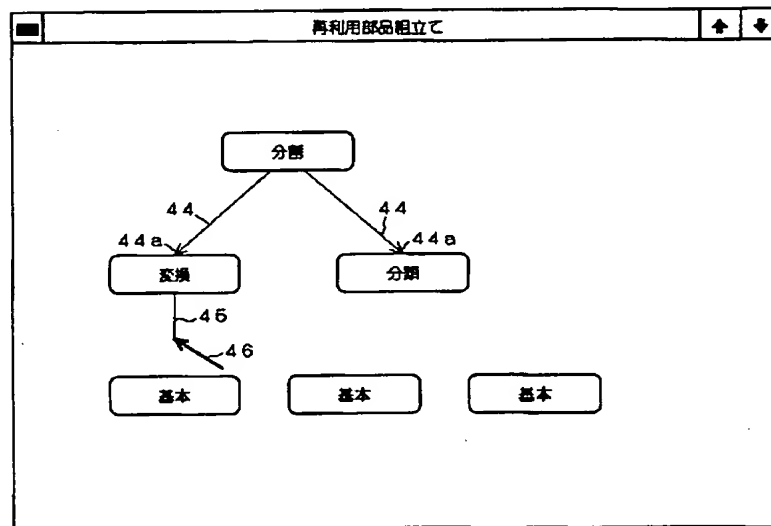
【図 4】



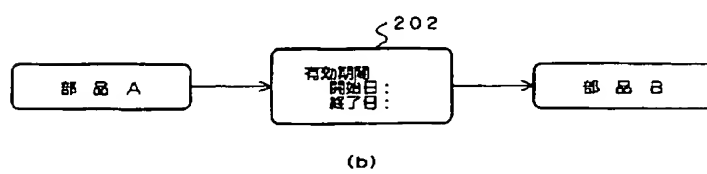
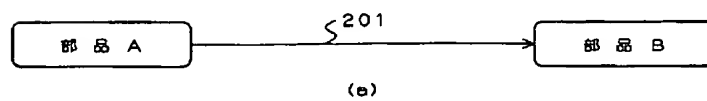
【図 5】



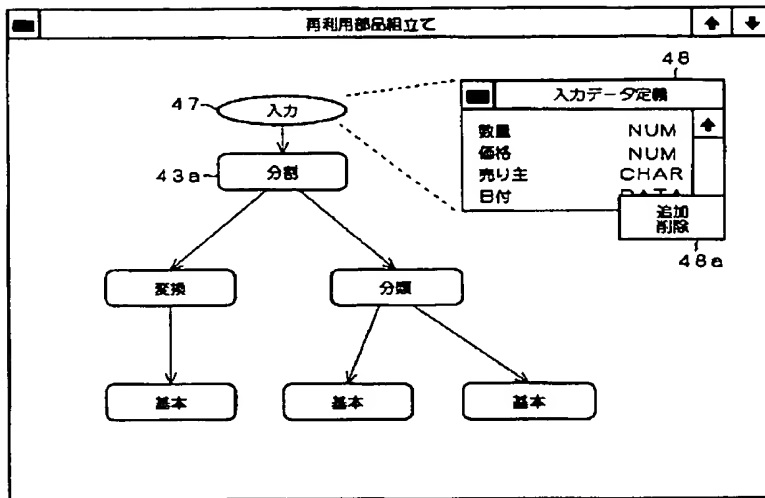
【図 6】



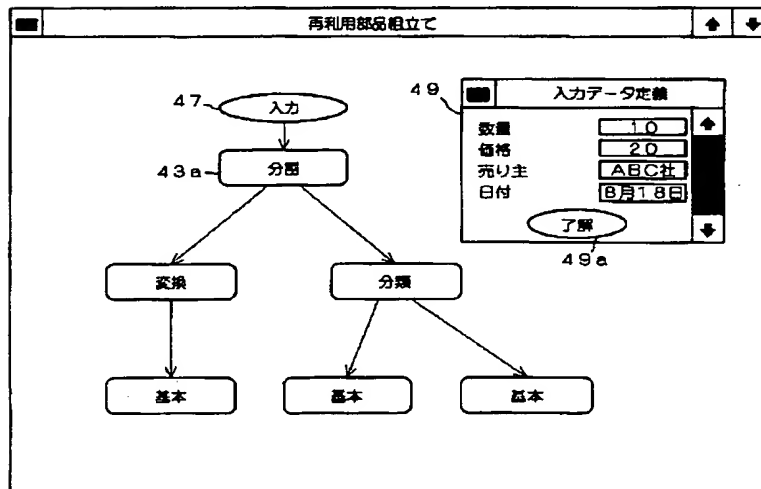
【図 20】



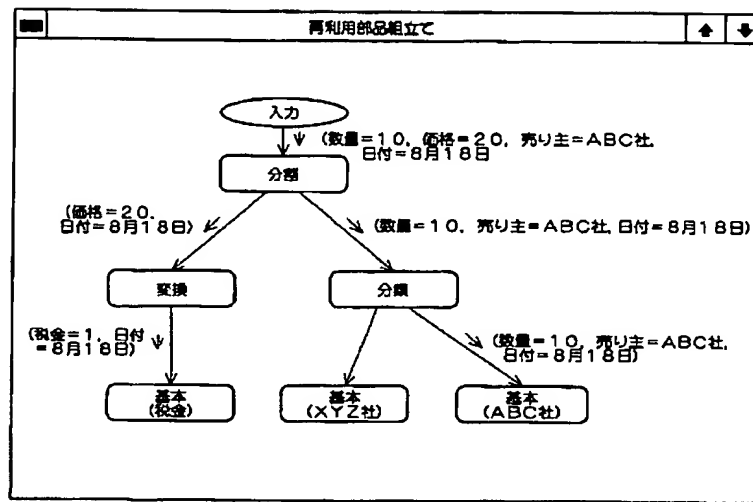
【図7】



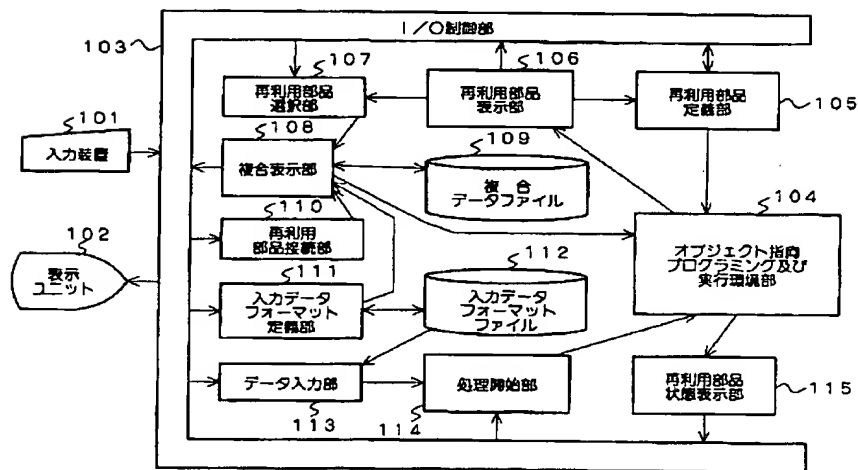
【図8】



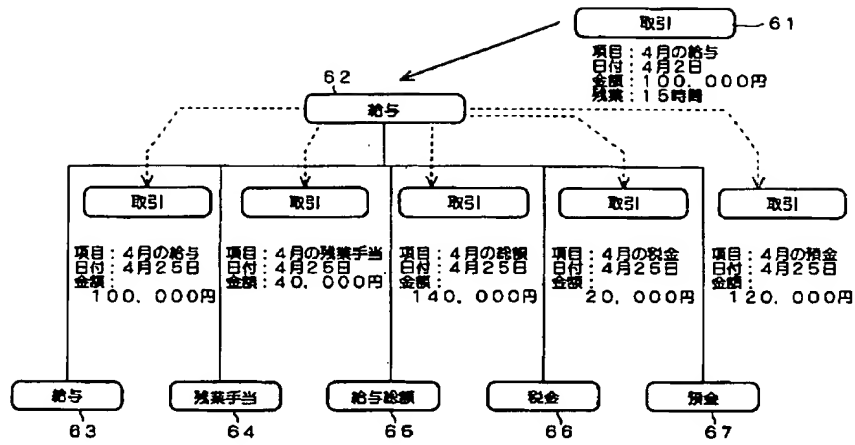
【図 9】



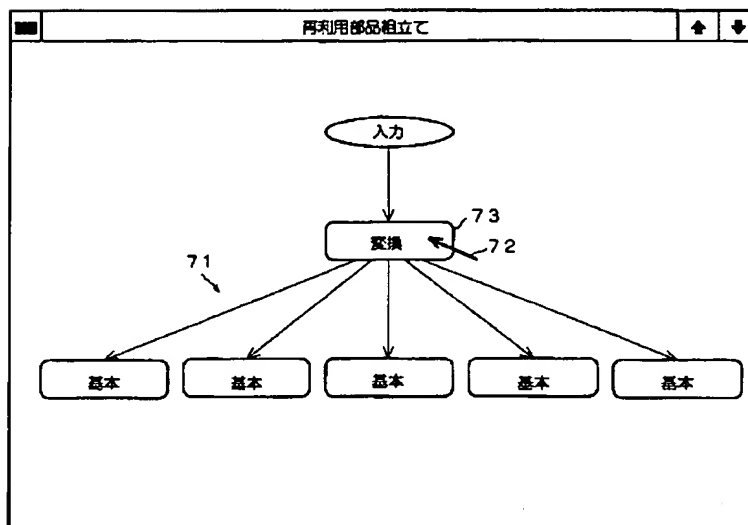
【図 10】



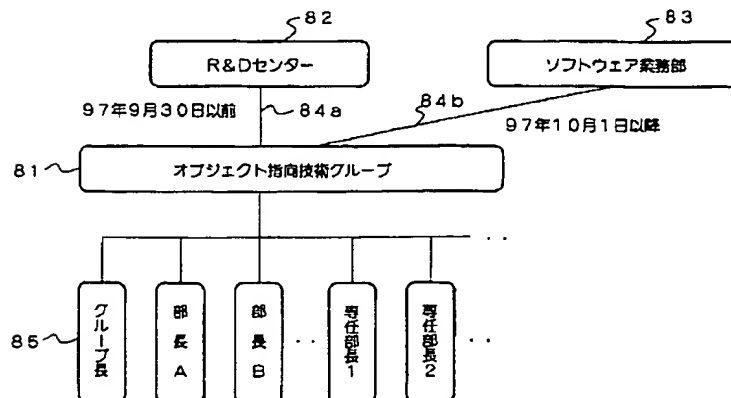
【図11】



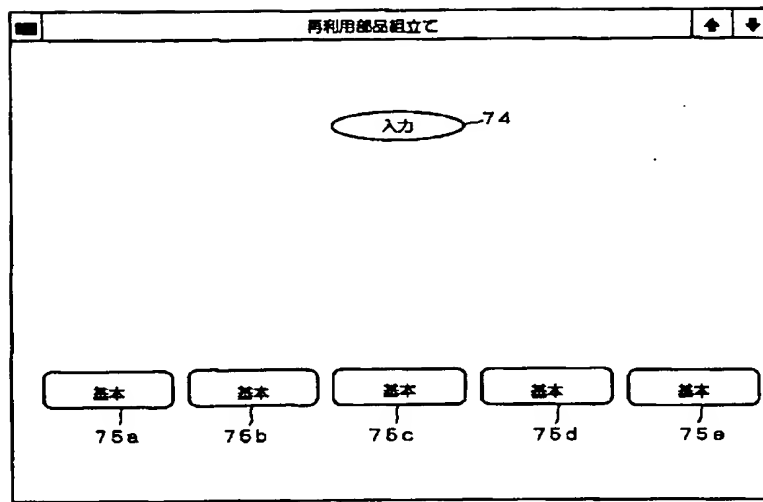
【図12】



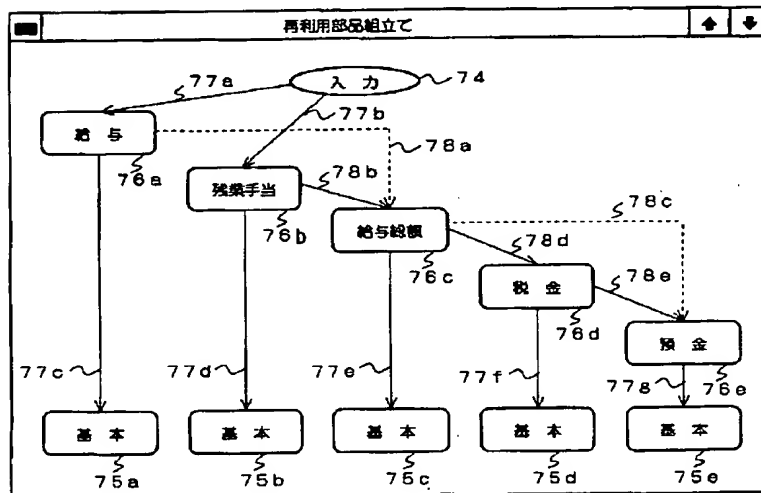
【図15】



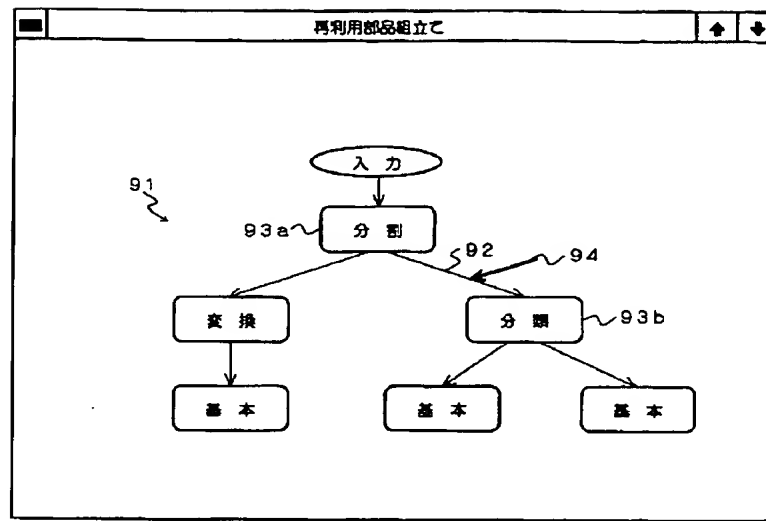
【図 1 3】



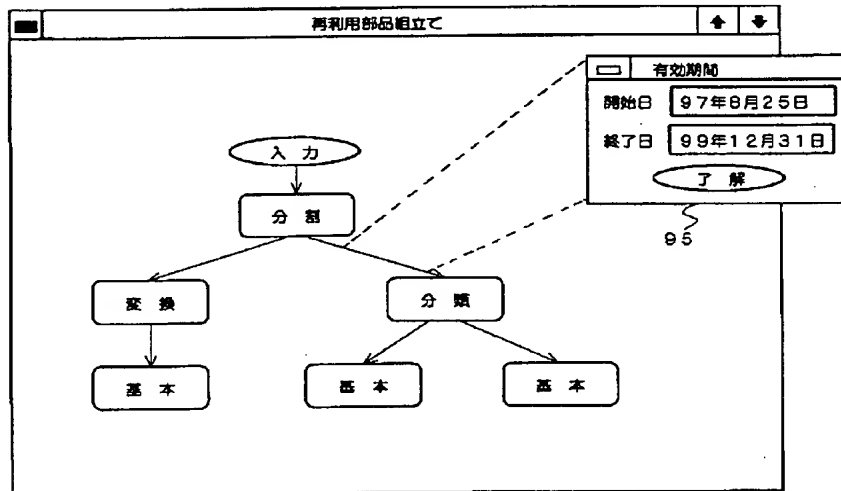
【図 1 4】



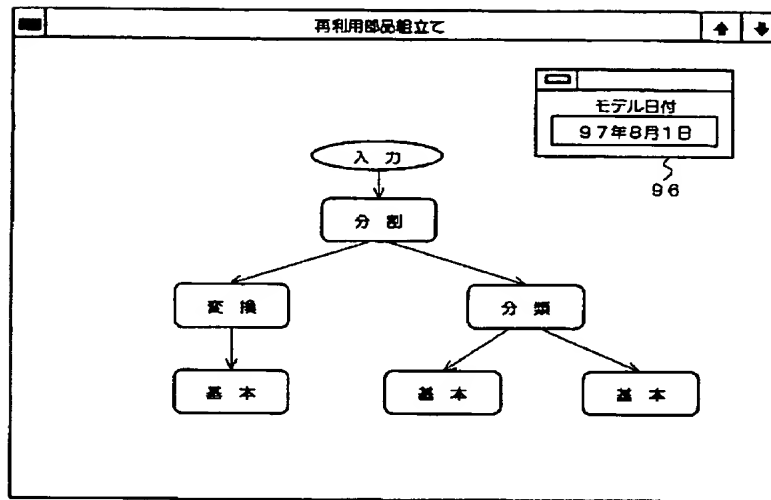
【図16】



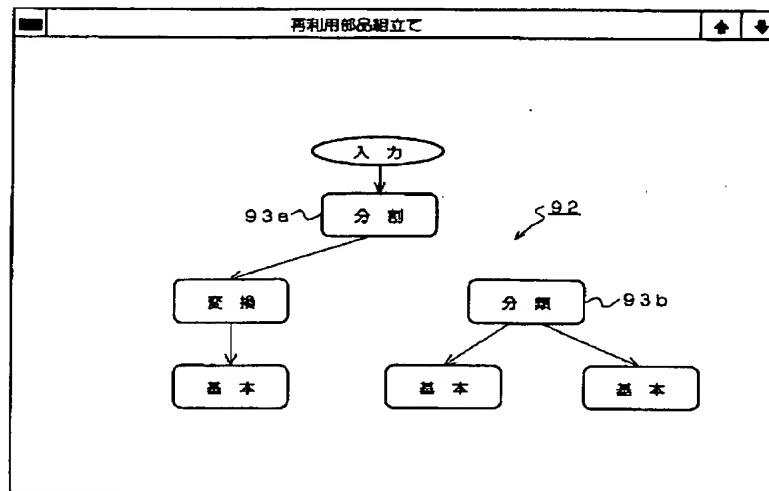
【図17】



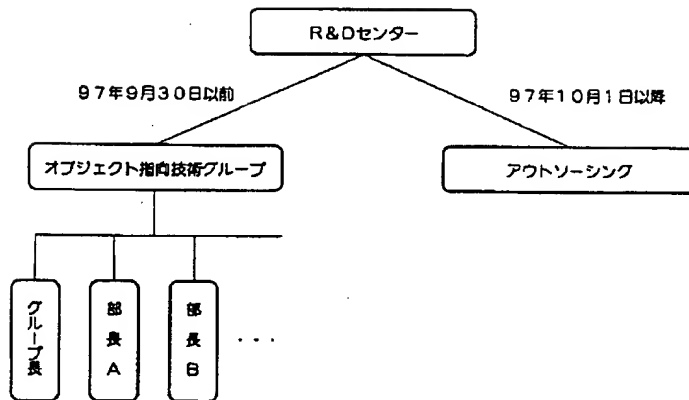
【図18】



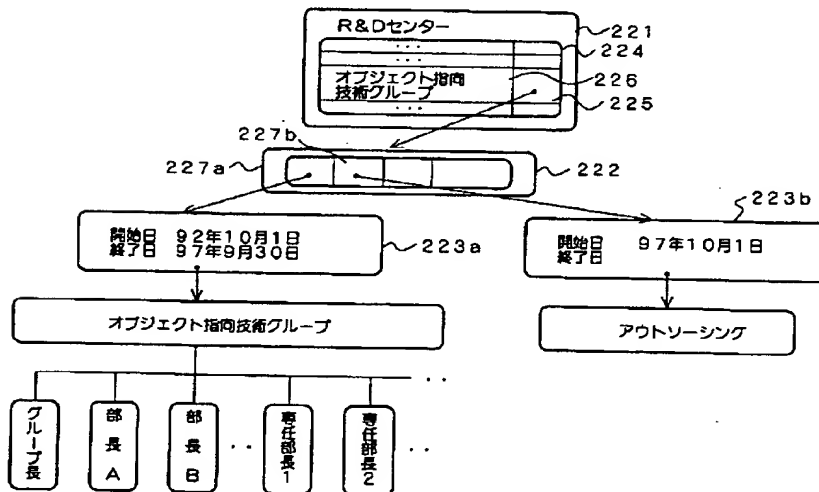
【図19】



【図21】



【図22】



【図23】

